Semantic Relation Classification via Hierarchical Recurrent Neural Network with Attention

Minguang Xiao **Cong Liu***
School of Data and Computer Science, Sun Yat-sen University
xiaomg@mail2.sysu.edu.cn, liucong3@mail.sysu.edu.cn

Abstract

Semantic relation classification remains a challenge in natural language processing. In this paper, we introduce a hierarchical recurrent neural network that is capable of extracting information from raw sentences for relation classification. Our model has several distinctive features: (1) Each sentence is divided into three context subsequences according to two annotated nominals, which allows the model to encode each context subsequence independently so as to selectively focus as on the important context information; (2) The hierarchical model consists of two recurrent neural networks (RNNs): the first one learns context representations of the three context subsequences respectively, and the second one computes semantic composition of these three representations and produces a sentence representation for the relationship classification of the two nominals. (3) The attention mechanism is adopted in both RNNs to encourage the model to concentrate on the important information when learning the sentence representations. Experimental results on the SemEval-2010 Task 8 dataset demonstrate that our model is comparable to the state-of-the-art without using any hand-crafted features.

1 Introduction

Semantic relation classification is an important task in natural language processing, which has attracted great attention in recent years. The goal is to identify the semantic relationship between a pair of nominals marked in a sentence. For instance, in the sentence "The software $[\text{company}]_{e_1}$ addressed the problem with the $[\text{publication}]_{e_2}$ of a fix on Saturday", the marked nominals of *company* and *publication* are of relationship *Product-Producer(e_2, e_1)*. Most conventional models focus on machine learning and feature design, which have been shown to obtain performance improvements (Kambhatla, 2004; Tratz and Hovy, 2010; Rink and Harabagiu, 2010).

Recently, neural network approaches have been widely used for relation classification, which aim at reducing the need of hand-crafted features. These approaches are broadly divided into two categories: one explores the effectiveness of using dependency paths and its attached subtrees between two nominals, and various neural networks are adopted to model the shortest dependency paths and dependency subtrees (Xu et al., 2015a; Xu et al., 2015b; Liu et al., 2015); the other exploits deep neural networks to learn syntactic and semantic features from raw sentences (Zeng et al., 2014; Dos Santos et al., 2015; Zhang et al., 2015), which has been proved effective, but inevitably suffers from irrelevant parts. Our paper introduces an attentive neural network that selectively focuses on useful information on raw sentences.

Context information of the annotated nominals has been widely believed to be useful for relation classification (Zhang et al., 2015; Thang Vu et al., 2016). In this work, we further explore the effectiveness of context information around the annotated nominals in a sentence. In our model, a sentence with two marked nominals is divided into three context subsequences according to two marked nominals: the left context subsequence, the middle context subsequence and the right context subsequence. This method is similar to Pei et al. (2015) and Thang Vu et al. (2016), which have showed that contextual information is effectively obtained by deep learning techniques. Instead of combining the middle context

*Cong Liu is the corresponding author.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1254–1263, Osaka, Japan, December 11-17 2016.

subsequence with the left and right context subsequences, respectively, as in (Thang Vu et al., 2016), we propose to learn context representations via recurrent neural networks that work on each context subsequence independently. For example, the sentence "The software $[\text{company}]_{e_1}$ addressed the problem with the $[\text{publication}]_{e_2}$ of a fix on Saturday" is split into three subsequences: " The software", "addressed the problem with the" and "of a fix on Saturday". And the marked nominals of $[\text{company}]_{e_1}$ and $[\text{publication}]_{e_2}$ are not included in any context subsequence. As a result, the sentence is divided into five parts: three context subsequences and two annotated nominals. Our sentence representations are leant hierarchically from context subsequences to sentences using a hierarchical recurrent neural network, which firstly learns the context representation of each context subsequence independently, and then encodes the semantics of context subsequences into a sentence representation for the relation classification. Furthermore, we introduce the attention mechanism (Bahdanau, 2014; Rush, 2015; Rocktäschel et al., 2016) that encourages the model to focus on the important information. Experimental results demonstrate that our model is comparable to the state-of-the-art with a single model that works on the raw sentences.

In the rest of this paper, we review recurrent neural networks in Section 2. We provide details about our model in Section 3. Section 4 presents our experiments and their results. Finally, we make a conclusion in Section 5.

2 Recurrent Neural Networks

Recurrent neural networks (RNNs) (Elman, 1990; Mikolov et al., 2010) project a sequence of inputs $x_1, \ldots, x_T$ to a sequence of outputs $y_1, \ldots, y_T$ via an affine transformation followed by a non-linear function. At timestep *t*, a standard RNN computes the new hidden vector as

$$h_t = f(Wx_t + Uh_{t-1} + b) \tag{1}$$

where *W* is trained matrix transforming the current input x_t into the current state linearly, *U* is also trained matrix connecting the previous state h_{t-1} with the current state, and *b* is a bias term, and *f* is a non-linear function (e.g., tanh).

However, RNNs with the above form may suffer from gradient *exploding* or *vanishing* problem (Bengio et al., 1994; Hochreiter, 1997) during training when it is trained with the backpropagation through time algorithm (Rumelhart et al., 1986; Werbos, 1990; Williams and Zipser, 1995). To address this problem, long short-term memory network (LSTM) was proposed in (Hochreiter and Schmidhuber, 1997) where the architecture of a standard RNN was modified to avoid vanishing or exploding gradients. Many LSTM variants have been proposed, and here we adopt the version of Zaremba and Sutskever (2014a).

The LSTM model comprises a memory cell that can store information over a long period of time, and three gates that allow it to control the flow of information into and out of the cell: input gate, forget gate, and output gate. Concretely, the LSTM unit at time step *t* encompasses a collection of vectors: an input gate i_t, a forget gate f_t, an output gate o_t, a memory cell c_t, and a hidden state h_t. The unit accepts an input vector x_t, the previous hidden state h_{t-1}, and the memory cell c_{t-1} and computes the new vectors using the following equations:

$$\begin{aligned}
i_t &= \sigma(W^{(i)}x_t + U^{(i)}h_{t-1} + b^{(i)}) \\
f_t &= \sigma(W^{(f)}x_t + U^{(f)}h_{t-1} + b^{(f)}) \\
o_t &= \sigma(W^{(o)}x_t + U^{(o)}h_{t-1} + b^{(o)}) \\
u_t &= tanh(W^{(u)}x_t + U^{(u)}h_{t-1} + b^{(u)}) \\
c_t &= i_t \odot u_t + f_t \odot c_{t-1} \\
h_t &= o_t \odot tanh(c_t)
\end{aligned} \tag{2}$$

where σ denotes the element-wise application of the logistic function, $\odot$ denotes the element-wise multiplication of two vectors, *W* and *U* are weight matrices, and *b* are bias vectors.

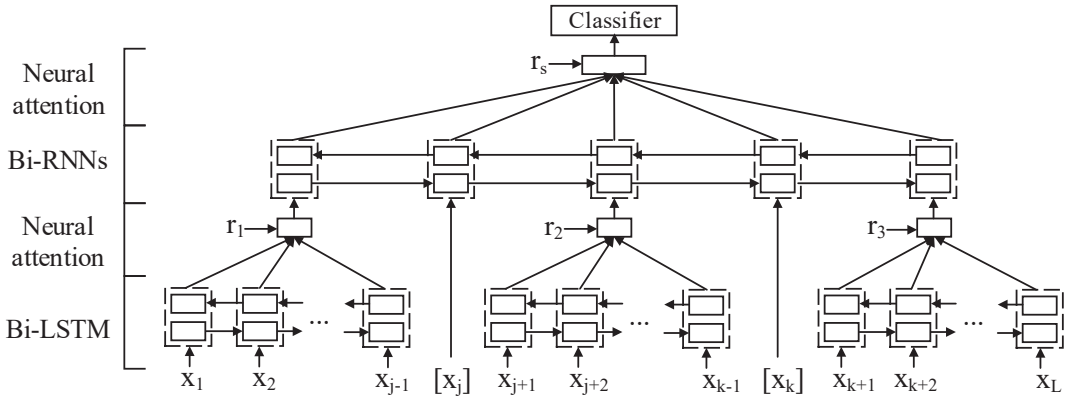

Figure 1: The architecture of our model. Given a sentence consisting of *L* words, it is divided into the left context subsequence $[x_1, \ldots, x_{j-1}]$, the middle context subsequence $[x_{j+1}, \ldots, x_{k-1}]$ and the right context subsequence $[x_{k+1}, \ldots, x_L]$. $[x_j]$ and $[x_k]$ represent the marked nominals e_1 and e_2 respectively.

3 Model

In this section, we introduce the proposed neural model that learns distributed representations from raw sentences. These representations serving as features are further used for relation classification. An overview of our model is shown in Figure 1.

Given a sentence with two annotated nominals, the sentence is firstly divided into five parts (three context subsequences and two annotated nominals) based on the two marked nominals (Section 3.1). Next, the model computes the distributed representations for the context subsequences using a bidirectional LSTM that works on word vectors (Section 3.2). Lastly, these distributed context representations are further encoded into a sentence representation via a bidirectional RNN (Section 3.3). Furthermore, we extend this model with a neural attention that encourages the model to focus on important information.

3.1 Context Subsequences

In most cases different contexts have different functions for the meaning of sentences. Some recent work fell into the idea that the middle context contains the most relevant information for relation classification, combining the middle context with the left and right context respectively (Zhang et al., 2015; Thang Vu et al., 2016). We instead model each context part independently, which allows the model to automatically identify contexts that contain useful information.

Given a sentence *s* and its annotated nominals e_1 and e_2, the sentence first is split into five parts according to the two annotated nominals: the left context subsequence, entity e_1, the middle context subsequence, entity e_2 and the right context subsequence. Preprocessing the sentence in such a way allows the model to encode each context subsequence independently.

3.2 Context Subsequence Composition

3.2.1 Word Encoder

A bidirectional LSTM (Bi-LSTM) (Graves and Schmidhuber, 2005; Graves et al., 2013) is applied to independently encoding each of the three context subsequences. A bidirectional LSTM consists of two LSTMs: the forward and backward LSTMs. They are run in parallel: the forward LSTM inputs the words from x_1 to x_T, and the backward LSTM inputs in an reverse order from x_T back to x_1. At time step *t*, we obtain the hidden state (denoted as h_t) of the bidirectional LSTM by concatenating the forward hidden state (denoted as $\overrightarrow{h_t}$) and the backward one (denoted as $\overleftarrow{h_t}$), i.e., $h_t = [\overrightarrow{h_t}, \overleftarrow{h_t}]$. Bi-LSTM can summarize the information from the whole context subsequence centered around words, which let the model understand the meaning of words comprehensively.

Given a sentence *s* divided into the left context subsequence c_1, the middle context subsequence c_2 and the right context subsequence c_3, we assume that the sentence *s* contains *L* words and the context subsequence c_i has T_i words, where *i* $\in [1, 3]$. The input to Bi-LSTM is a context subsequence c_i: [x_{i1}, ..., x_{iT_i}] where x_{it} is the word vector for word w_{it}. At time step *t*, the encoder produces a hidden state h_{it} which gathers the information of the whole context subsequence c_i centered around w_{it}. The equations are following:

$$\overrightarrow{h_{it}} = \overrightarrow{LSTM}(x_{it}) \tag{3}$$

$$\overleftarrow{h_{it}} = \overleftarrow{LSTM}(x_{it}) \tag{4}$$

$$h_{it} = concat(\overrightarrow{h_{it}}, \overleftarrow{h_{it}}) \tag{5}$$

where *concat* is concatenation function, i.e., h_{it} = [$\overrightarrow{h_{it}}$; $\overleftarrow{h_{it}}$].

Note that our model encodes the left, middle and right context subsequence independently but with one Bi-LSTM. [1]

3.2.2 Word-level Attention

Due to the fact that raw sentences contain more information than the shortest dependency paths, there may be some irrelevant information in raw sentences. For concentrating on these words that are important to predict the relationship of entities, it can be a good strategy to pay more attention on these words. To encourage such behavior, this paper introduces a word-level attention mechanism. The attention mechanism enables the model to differently attend over the hidden vectors of Bi-LSTM along a context subsequence, and produces a weighted representation m_i of them as follows:

$$\begin{aligned} z_{it} &= tanh(W^{(w)} h_{it} + W^{(c)} r_i + b) \\ \alpha_{it} &= \frac{exp(v_z^\top z_{it})}{\sum_{j=1}^{T_i} exp(v_z^\top z_{ij})} \\ m_i &= \sum_{t=1}^{T_i} \alpha_{it} h_{it} \end{aligned} \tag{6}$$

where $W^{(w)}$ and $W^{(c)}$ are weight matrices, *b* is a bias vector, v_z is a weight vector and $v_z^\top$ is its transformation, and r_i is an external context vector that is randomly initialized and jointly optimized during training.

The attention representation z_{it} corresponding to the *t*-th word w_{it} in the context subsequence c_i is computed via a non-linear combination of the hidden state h_{it} and the external context vector r_i. The attention weight α_{it} for the *t*-th word w_{it} in the context subsequence c_i is a probability that is the normalized weight of z_{it} (parameterized by v_z) through a softmax layer, reflecting the importance of the *t*-th word w_{it} with respect to the meaning of the context subsequence c_i in classifying the relationship of two entities. The external context vector r_i not only represents the high-level meaning of the context subsequence c_i, but also allows the model to identify that the word w_{it} is in the context subsequence c_i.

3.3 Sentence Composition

After establishing an attention-based Bi-LSTM (Section 3.2) to capture the meaning of three context subsequences, resulting in three context representations, there is one difficulty that how to further obtain the semantic composition of these context representations plus two representations of marked nominals. Note that there are five semantic representations. The most common approach is that a multilayer perceptron (MLP) is adopted to take these representations as input and compute semantic compositionality

[1] We adopt Bi-LSTM to encode each context subsequence separately even if it contains few words, such as one word.

for them. In this work, we adopt a Bi-RNN to integrate syntactics and semantics of three context subsequences and two annotated nominals into sentence representation *s*, which is further fed into a classifier for relation classification. We propose to learn sentence representations via Bi-RNNs for two reasons: (1) a sentence containing two annotated nominals divided into three context subsequences that are ordered as in the sentence, can be treated as a short sequence that consists of five tokens; (2) recurrent neural networks are competent enough to model the semantics of these context subsequences and their inherent relations, which is important to obtain the semantic meaning of the sentence. The experimental results demonstrate that Bi-RNNs significantly outperform MLP.

Let Y be a matrix containing five column vectors [m_1, m_{e_1}, m_2, m_{e_2}, m_3], where m_i ($i \in [1, 3]$) is the representation of the context subsequence c_i, and m_{e_1} and m_{e_2} are the representations of annotated nominals e_1 and e_2. [2] To obtain compositional vector representations for sentences, we iterate the following sequence of equations:

$$\overrightarrow{h_j} = \overrightarrow{RNN}(y_j) \tag{7}$$

$$\overleftarrow{h_j} = \overleftarrow{RNN}(y_j) \tag{8}$$

$$h_j = concat(\overrightarrow{h_j}, \overleftarrow{h_j}) \tag{9}$$

where $y_j \in Y$ (j $\in [1, 5]$) is the *j*-th column vector in Y.

Note that the sentence only contains five elements, our model do not make any assumptions about the type of RNNs used in this subsection. But as far as comparison goes, LSTMs performs better than the standard RNNs.

To selectively focus on the important context subsequences, it is an alternative solution to applying neural attention to the hidden vectors of the above Bi-RNNs, similar to Subsection 3.2.2. We also make further extensions such as average pooling and max pooling.

3.4 Training

A fully connected softmax layer is used as classifier for classification. It produces the probability distribution *p* over relation types conditioned on the sentence representation *s*:

$$p = softmax(W^{(s)}s + b^{(s)}) \tag{10}$$

The training objective is to minimize the cross-entropy error between the ground truth and predicted label. The parameters of our model are optimized using AdaGrad (Duchi et al., 2011) with a learning rate of 0.01, a mini-batch size of 5 and a L_2 regularization coefficient of 10^{-6}. The details are described further in Section 4.2.

4 Experiments and Evaluation

4.1 Dataset

In our experiments, we evaluate our model on the SemEval-2010 Task 8 dataset (Hendrickx et al., 2010), which is one of the most widely used benchmarks for relation classification. The dataset contains 10,717 annotated sentences divided into 8,000 sentences for training and 2717 for testing. Each sentence is annotated with each of nine different relationship and an artificial relation *Other*, and each relationship has two direction except for the undirected relation *Other*. The nine directed relations are *Cause-Effect, Instrument-Agency, Product-Producer, Content-Container, Entity-Origin, Entity-Destination, Component-Whole, Member-Collection, and Message-Topic*.

The official evaluation metric is the macro-averaged F1-score (excluding *Other*), and takes into consideration the directionality. We use the official scorer to test the model performance.

[2] We use an additional *tanh* layer to map the word vectors of annotated nominals e_1 and e_2 to the dimensionality of the hidden size of the Bi-LSTM.

4.2 Implementation

We tune the hyperparameters for our model using 5-fold cross-validation. We pretrain 200-dimensional word embeddings using *word2vec* (Mikolov et al., 2013) on the English Wikipedia corpus, and randomly initialize other hyperparameters. We set the LSTM dimension to be 200. We apply dropout only on the word embeddings and outputs of LSTM as in (Zaremba et al., 2014b), and the dropout rate is 0.2.

To enable a direct comparison with the previous work, we use the same features: position features, WordNet hypernyms and NER. WordNet hypernyms and NER were obtained using the tool of Ciaramita and Al-tun (2006). [3]

4.3 Results

Model	features	F1
Bi-LSTM + MLP	-	82.43
	+ all features	83.30
Bi-LSTM + Bi-RNN	-	82.67
	+ all features	82.92
Bi-LSTM + Bi-LSTM	-	83.90
	+ all features	84.27

(a) The effect of neural network architectures.

Method	features	F1
Concatenation	-	79.66
	+ all features	80.56
Average	-	79.91
	+ all features	81.39
Max-Pooling	-	81.67
	+ all features	82.48
Attention	-	83.90
	+ all features	84.27

(b) Comparison of different methods

Table 1: (a) F1-scores on the test data for various neural network architectures. We also test these models with three features of position features, WordNet and NER. (b) The comparison of different methods on SemEval-2010 Task 8 test set. Here the neural network architecture is the combination of two Bi-LSTMs.

4.3.1 The Effect of Different Components

The effect of neural network architectures We first analyze the effect of different neural network architectures of the combinations of Bi-LSTM with MLP, a standard Bi-RNN and Bi-LSTM separately. Here we apply neural attention to the hidden states of RNNs (Bi-LSTM and Bi-RNN). To ensure the number of parameters comparable, we adopt a two-layer full-connected neural network with the hidden size of 600 dimension and a non-linear function of *tanh* to serve as MLP. And the hidden size of the standard RNN is 350-dimensional. From Table 1a, we find that both the combinations of Bi-LSTM with Bi-RNN and Bi-LSTM outperform the combination of Bi-LSTM and MLP without any features. In particular, the combination of Bi-LSTM and Bi-LSTM achieves the best result 83.90% without any feature, and its F1-score is about 1.5% higher than the model of Bi-LSTM+MLP. The results indicate that the neural architecture of two Bi-LSTMs effectively captures semantic meanings of these context subsequences and their inherent relations, and obtains more robust sentence representations for relation classification. In this paper, we tackle the relation classification task using the combination of two Bi-LSTMs.

The comparison of different methods Table 1b shows experiments for our model with various methods for the hidden vectors of Bi-LSTMs. We begin with the model using the concatenation of the final state of forward and backward LSTMs. And then we replace concatenation operation with average pooling, max-pooling and neural attention respectively. Not surprisingly, processing the hidden vectors of Bi-LSTMs via neural attention achieves the best result, which gives an improvement of 2.23 percentage points in F1-score over max-pooling. We suspect that this is due to the attention model being run in a more focused way that makes it easier to capture large important information from contexts. We also consider the impact of features for these methods. Results in Table 1b show that by adding features the F1-scores of all methods improve, which hints that three features are useful for relation classification.

[3] sourceforge.net/projects/supersensetag/

Model	Feature Set	F1
SVM	+POS, WordNet, prefixes and other morphological features, dependency parse, Levin classes, PropBank, FanmeNet, NomLex-Plus, Google n-gram, paraphrases, TextRunner	82.2
MV-RNN	- +POS, NER, WordNet	79.1 82.4
FCM	- +dependency parsing, NER	80.6 83.0
CNN	- +position features, words around nominals, WordNet	69.7 82.7
BLSTM	- +POS, NER, WordNet, position features, dependency feature, relative-dependency feature	82.7 84.3
DepNN	+WordNet +NER	83.0 83.6
SDP-LSTM	- +POS embeddings, WordNet embeddings, grammar relation embeddings	82.4 83.7
depLCNN + NS	- +WordNet, words around nominals	84.0 **85.6**
Our model	- +position features, WordNet, NER	83.9 84.3

Table 2: Experimental results of our model against other models.

4.3.2 Comparison with State-of-the-art Models

Table 2 compares our model with several start-of-art models. The SVM model (Rink and Harabagiu, 2010) is used for relation classification by combining lexical and semantic features. It extracts these hand-crafted features from sentences with the use of many external resources. Socher et al. (2012) extend the recursive neural networks with matrix-vector spaces (MV-RNN), and use MV-RNN to learn representations along the constituency tree for relation classification. Yu et al. (2014) propose factor-based compositional embedding models (FCM) for relation classification. It learns representations for the substructures of an annotated sentence, which are further used for classification. Zeng et al. (2014) exploit a convolutional neural network (CNN) to extract lexical and sentence level features for relation classification. And they design position features to specify the target nouns in the sentence, which leads to better performance for their model. CR-CNN outperforms the state-of-art by using a new ranking loss function and omitting the representation of the *Other* class for diminishing its effect, as proposed by (Dos Santos et al., 2015). Zhang et al. (2015) utilized bidirectional LSTMs (BLSTM) to capture the sentence level features and concatenated them and lexical level features to form the finally feature vector for relation classification. Liu et al. (2015) design a dependency-based framework (DepNN) to learn semantic representations of the augmented dependency paths that are the combination of the shortest dependency paths and their dependency subtrees. Xu et al. (2015a) build multiple LSTMs to model the different channels of word vectors, POS, grammatical relations, and WordNet along the shortest dependency paths and achieves an F1-score of 83.7 (SDP-LSTM). Xu et al. (2015b) propose to learn a robust representation using a convolutional neural network that works on the dependency path between subjects and objects, and propose a negative sampling strategy (NS) to address the relation directionality (DepLCNN). Thang Vu et al. (2016) design extended middle context and present a new context representation for convolutional neural networks for relation classification (ER-CNN). And they also propose connectionist bi-directional recurrent neural networks (R-RNN) that adds a connection to the hidden states of bi-directional recurrent neural networks.

We observe in Table 2 that our model is comparable to the state-of-the-art (previous best result is 84.0% obtained by depLCNN + NS) without any features, whereas depLCNN works on the shortest dependency

Model	Feature Set	F1
CR-CNN	-	82.8
	+position features	84.1
R-RNN	+position features, position indicators, entity flag	83.4
ER-CNN	+position features, extended middle context	84.2
ER-CNN + R-RNN	+all features, voting scheme	84.9
Our model	-	84.1
	+position features	84.5

Table 3: Comparison of ranking models (no lexical features).

paths, which consist of most relevant information and avoid negative effect from irrelevant parts in the sentences. This result suggests that our model automatically focuses on important information related to determining the relationship of two entities. The F1-score is improved by adding three features but not as obvious as in (Zeng et al., 2014; Xu et al., 2015b) (CNN, depLCNN). We argue that this is due to Bi-LSTMs being able to learn position information on sequences and lexical features leading to overfitting as in (Yu et al., 2014; Liu et al., 2015).

4.3.3 Comparison of ranking models

For fair comparison, we also replace the softmax layer with a ranking layer to train our model, as proposed in (Dos Santos et al., 2015). We use training settings following Thang Vu et al. (2016). More details about ranking layer are described in (Dos Santos et al., 2015; Thang Vu et al., 2016).

From Table 3, we observe that our model outperform the state-of-the-art without any feature, whereas previous work's best reported performance is 83.9% in ER-CNN using word embeddings of size 400. Combining ER-CNN and R-RNN using a voting scheme achieves a state-of-the-art result of 84.9 in F1-score, which is presented by (Thang Vu et al., 2016). But our model reaches a new state-of-the-art result with a single model when position features are added, and outperforms the model of ER-CNN that learns context representations for two contexts of the combinations of the middle context with the left and right context respectively.

5 Conclusion

In this work, we introduce a hierarchical recurrent neural network model that learns useful features from raw sentences for relation classification. We further extend the model with neural attention at two different levels that provides significant improvements over the concatenation, average pooling and max-pooling. Our model shows comparable performance to the state-of-the-art on the SemEval-2010 Task 8 dataset without using any costly hand-crafted features. In addition, the models presented here are general hierarchical models, and are therefore suitable for hierarchical structures, such as paragraphs and documents.

Acknowledgements

This work was partially supported by National Science Foundation of China (grant 61472459). We thank the anonymous reviewers for their insightful comments.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks 5(2):157–166*.

John C. Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research, 12:2121–2159*.

Jeffrey L. Elman. 1990. Finding structure in time. *Cognitive Science, 14(2):179–211*.

Alex Graves, Navdeep Jaitly, and A-R Mohamed. 2013. Hybrid speech recognition with deep bidirectional LSTM. *In IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU). pages 273–278*.

A. Graves and J. Schmidhuber. 2005. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Net- works, 18(5):602–610*.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, PreslavNakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2010. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. *In Proceedings of the 5th International Workshop on Semantic Evaluation, pages 33–38. Association for Computational Linguistics*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation, 9(8):1735–1780, November*.

Sepp Hochreiter. 1998. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 6(02):107–116*.

Nanda Kambhatla. 2004. Combining lexical, syntactic, and semantic features with maximum entropy models for extracting relations. *In Proceedings of the ACL 2004 on Interactive Poster and Demonstration Sessions, page 22. Association for Computational Linguistics*.

Yang Liu, Furu Wei, Sujian Li, Heng Ji, Ming Zhou, Houfeng Wang. 2015. A Dependency-Based Neural Network for Relation Classification. *InProceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Short Papers), pages 285–290. Association for Computational Linguistics*.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan "Honza" Černocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. *In Proceedings of the 11th Annual Conference of the International Speech Communication Association (INTERSPEECH 2010), pages 1045–1048*.

Tomáš Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *In Proceedings of the Workshop at ICLR*.

Cícero Nogueira Dos Santos, Bing Xiang, and Bowen Zhou. 2015. Classifying relations by ranking with convolutional neural networks. *In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, pages 626–634. Association for Computational Linguistics*.

Wenzhe Pei; Tao Ge; Baobao Chang. 2015. An Effective Neural Network Model for Graph-based Dependency Parsing. *In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, pages 313–322. Association for Computational Linguistics*.

Bryan Rink and Sanda Harabagiu. 2010. Utd: Classifying semantic relations by combining lexical and semantic resources. *In Proceedings of the 5th International Workshop on Semantic Evaluation, pages 256–259. Association for Computational Linguistics*.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiský, Phil Blunsom. 2016. Reasoning about Entailment with Neural Attention. *In Proceedings of ICLR2016*.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. 1986. Learning Internal Representations by Error Propagation. *In: J. L. McClelland, D. E. Rumelhart, and The PDP Research Group: "Parallel Distributed Processing, Volume 1: Foundations". The MIT Press*.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. *In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 379–389. Association for Computational Linguistics*.

M. Schuster and K. K. Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Sig- nal Processing, 45(11):2673–2681*.

Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. 2012. Semantic Compositionality through Recursive Matrix-Vector Spaces. *In Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pages 1201–1211. Association for Computational Linguistics*.

Ngoc Thang Vu, and Heike Adel and Pankaj Gupta and Hinrich Schütze. 2016. Combining Recurrent and Convolutional Neural Networks for Relation Classification. *In Proceedings of the 15th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Stephen Tratz and Eduard Hovy. 2010. Isi: automatic classification of relations between nominals using a maximum entropy classifier. *In Proceedings of the 5th International Workshop on Semantic Evaluation, pages 222–225. Association for Computational Linguistics*.

Paul J. Werbos. 1990. Backpropagation through time: what it does and how to do it. *In Proceedings of the IEEE, 78(10):1550–1560*.

R. J. Williams and D. Zipser. 1995. Gradient-Based Learning Algorithms for Recurrent Networks and Their Computational Complexity. *In: Yves Chauvain and David E. Rumelhart: "Back-Propagation: Theory, Architectures and Applications". Lawrence Erlbaum Publishers*.

Yan Xu, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng, Zhi Jin. 2015a. Classifying Relations via Long Short Term Memory Networks along Shortest Dependency Paths. *In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1785–1794. Association for Computational Linguistics*.

Kun Xu, Yansong Feng, Songfang Huang and Dongyan Zhao. 2015b. Semantic Relation Classification via Convolutional Neural Networks with Simple Negative Sampling. *In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 536–540. Association for Computational Linguistics*.

Mo Yu, Matthew Gormley, and Mark Dredze. 2014. Factor-based compositional embedding models. *In Proceedings of the NIPS Workshop on Learning Semantics*.

Wojciech Zaremba and Ilya Sutskever. 2014a. Learning to execute. *arXiv preprint arXiv:1410.4615*.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014b. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. *In Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers, pages 2335–2344*.

Shu Zhang, Dequan Zheng, Xinchen Hu, Ming Yang. 2015. Bidirectional Long Short-Term Memory Networks for Relation Classification. *In Proceedings of the 15th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.